# Computer Vision and Imaging [06-30213]
# Assignment 1

9am, Monday, 8 March 2021

| | |
|---|---|
| Submission deadline: | **9am (GMT), Monday, 15 March 2021** |
| Instructor: | Dr. Hyung Jin Chang<br>Dr. Yixing Gao<br>Dr. Mohan Sridharan<br>Dr. Masoumeh Mansouri |
| Total marks: | 100 |
| Contribution to overall module mark: | 25% |
| Submission Method: | This assignment must be submitted through Canvas. |


| | |
|---|---|
| Name: | Selma Kander |
| Username: | sxk1093 |
| Study program: | BSc Computer Science |


## Part 1

**Question 1.1** *What affect does this kernel have? (Consider both this image and larger example images)*

This filter emphasises the edges of elements in an image. It highlights large differences in pixel values; a pixel whose neighbours have similar values will be darkened, and a pixel whose neighbour's values strongly differ will appear lighter. In this example, the biggest difference in value was between the centre column and its right-side neighbour: we can see that after applying the filter, this difference was emphasised. On a larger image, the end result would show a light outline of the defined features of an image, while the rest of the image will be dark.
Resulting convolved matrix:

$$\begin{bmatrix} -15 & -326 & 92 \\ -10 & -335 & 243 \\ -14 & -309 & 321 \end{bmatrix}$$

**Question 1.2** *What are their responses to uniform brightness? Are the filters isotropic or anisotropic and explain the terms. Which filter would you use for finding edges? What kind of edges would you use it to find?*

When using these filters on uniform brightness, these kernels would have no effect on the image. The final result would be the same as the initial one. An isotropic filter treats every axis the same, this means the values have to be the same in every direction; an anisotropic filter applies more emphasis on a certain axis or in a certain direction. The first filter is isotropic because it doesn't prioritise a specific direction, but is the same in every direction from the centre. The second filter is anisotropic, as the values are different depending on the direction from the centre(e.g: the top row is all '1's whereas the bottom row is all '-1's.

I would use the second filter for edge detection, as it emphasises the top edge and centre-bottom. With this filter I expect to find horizontal edges, as the contrast in values is between horizontal rows (e.g the top and second-top rows).

**Question 1.3** *Is it possible to use the intrinsic and extrinsic camera properties to transform from a point in the camera 2D pixel coordinate system to a point in the world 3D coordinate system, and why?*

Transforming a point in the camera 2D pixel coordinate system to a point in the world 3D coordinate system can be made challenging by the existence of the camera's intrinsic parameters: focal length, principal point, aspect ratio, angle between axes, etc. This means that one unit in the camera's coordinate system will differ from one unit in the surrounding world's coordinate system. A way to fix this is to link the camera's coordinate system to a fixed world coordinate system and specify its position and orientation in space using the extrinsic parameters. In order to project 3D coordinates onto a 2D plane, we use two consecutive projections: first, we project the 3D world coordinates onto the 3D camera coordinate system, and then we project this 3D camera coordinate system onto a 2D image plane. Therefore to transform a point in the camera 2D pixel coordinate system to a point in the world 3D coordinate system we do the opposite chain of operations: we first convert the image plane to the camera's 3D coordinate system.
The camera's 3D coordinate system is connected to the image plane by a pinhole system. The Principal axis is a line which links the camera centre to the Principal point on the image plane. The camera centre represents the origin of the normalised camera coordinate system. We change the coordinate system's origin so the bottom left corner becomes the centre, so as to reverse the operation made when 3D information is translated to 2D. We then convert the coordinates from a Cartesian coordinate system (with two elements) to a homogeneous coordinate system (with three elements), by adding a third element which takes the default value of '1'. This means that multiplying the coordinates by a non-zero scalar value will not change the point.
We then project these coordinates to a rectangular (or skewed) sensor with a size fixed by the manufacturer. We obtain a rectangular pixel matrix. To convert this to real world coordinates, we need to take into account the calibration matrix, projection centre in world, and rotation matrix of the camera. Some additional parameters may need to be taken into account, such as the radial distortion of an image caused by the lens, which means we need to remove non-linearity before applying a pinhole model. There are a total of 11 parameters to determine in order to transform a point in the camera 2D pixel coordinate system to a point in the world 3D coordinate system. We should note there could be some considerable loss of information, especially if there is only one image to construct the 3D information off of. One solution would be to take multiple photos of the same object from different points of view and stitch them together.

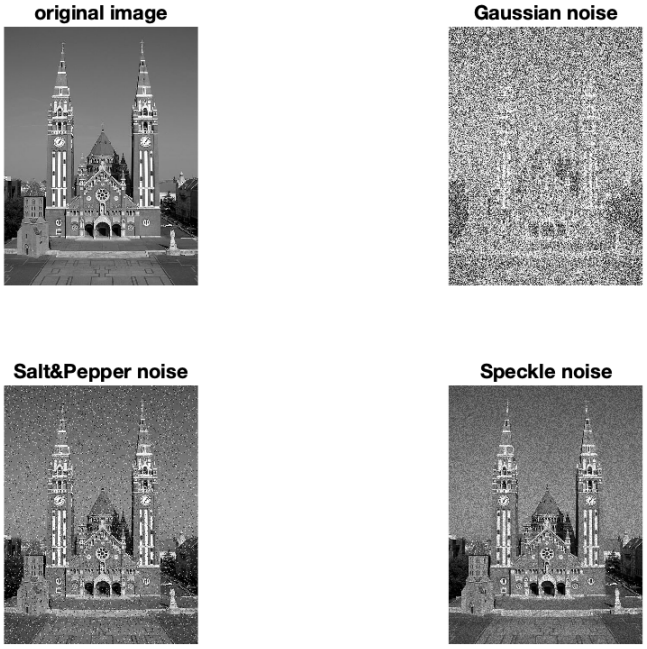
# Part 2

For this task, you will have to submit the following file:

- code **username_assignment1_part2.m**

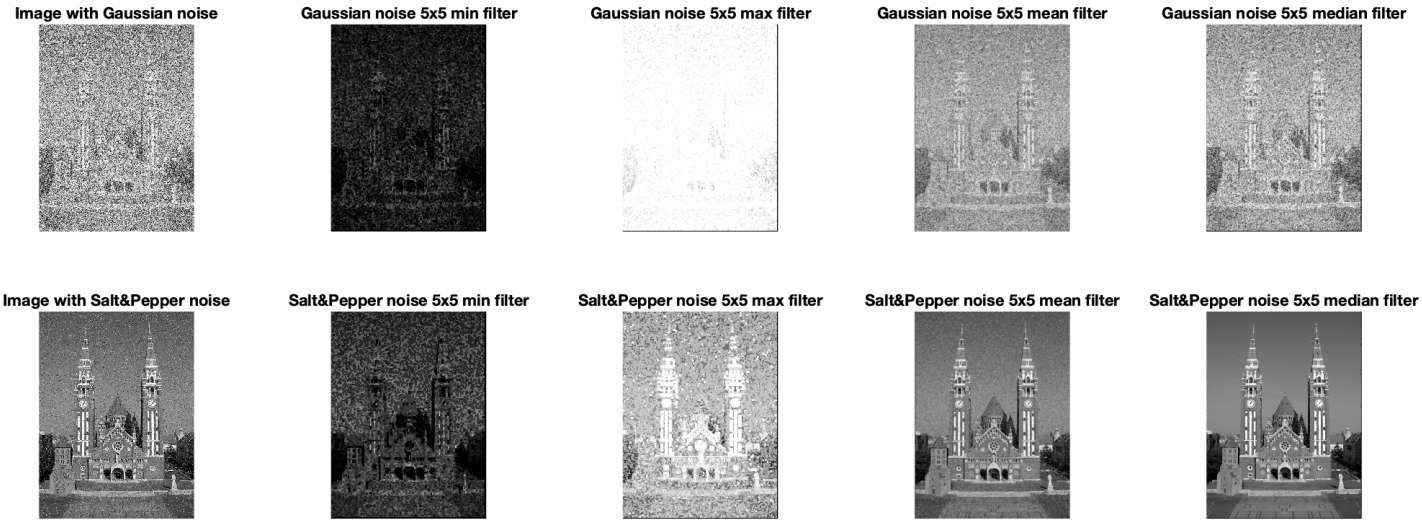
**Question 2.1.1** *Discuss your observations.*
We can see Gaussian noise had the strongest effect on the original image. This may be due to the parameters we chose. Salt Pepper, as well as Speckle noise had more subdued effects on the image. With the Gaussian noise, we can still discern some of the main color contrasts, such as the entrance of the building, but much detail is lost and some elements blend together, like parts of the towers with the sky for example. With the salt Pepper noise, the image's details are mostly preserved. This noise filter looks the most disturbing as it only adds purely white and purely black pixels, which doesn't

look natural. Finally, Speckle noise did the best job of preserving the image in my opinion. Although there is some unnatural variation in color, in the sky for example, it is not as disruptive as Salt  Pepper. The details of the image are also mostly well preserved.



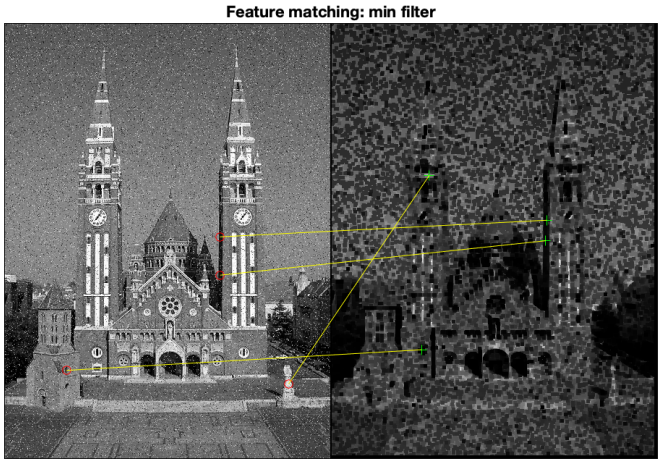

**Question 2.1.2** *How well did they work and why? Discuss your observations.*

The Maximum denoising filter did not work well. It effectively removed the light noise, but emphasized the dark noise. This makes sense because the filter makes each pixel as dark as its darkest neighbour. For Salt  Pepper, we can clearly see large squares in the place of the small specks of black in the original image.
The Minimum denoising filter did not work well either. Contrary to the maximum filter, it removed the dark noise but emphasized the light noise by replacing each pixel's value with it's lightest neighbour's value. We can see the effect it had on Salt  Pepper noise, where white features (like the details on the towers) were thickened. This shows the Max and Min filters are bad for denoising, as they also impacted the original photo itself.
The Mean denoising filter did a better job. It simply blurs the image, so this works well for the Salt  Pepper noise, which added harsh black and white pixels. For the Gaussian noise however, it didn't improve the appearance of the image, even though the blurriness makes the image less harsh to look at. Overall the Mean filter didn't 'add damage' to the images so it's better than the previous filters.
The Median denoising filter did the best job in my opinion. For the Salt  Pepper noise, it almost perfectly reversed back to the original image. Some specks are left, in the sky for example, and the lines appear slightly rougher than in the original, but no details have been erased. For the Gaussian noise, it restored some of the contrast which makes the details easier to discern, but overall doesn't have a strong impact. The image is still considerably noisy. I think the Median denoising filter worked better than Mean because it gives less importance to pixels with drastically high or low values. This means artificially added black and white filters, such as in Salt  Pepper, are largely erased.

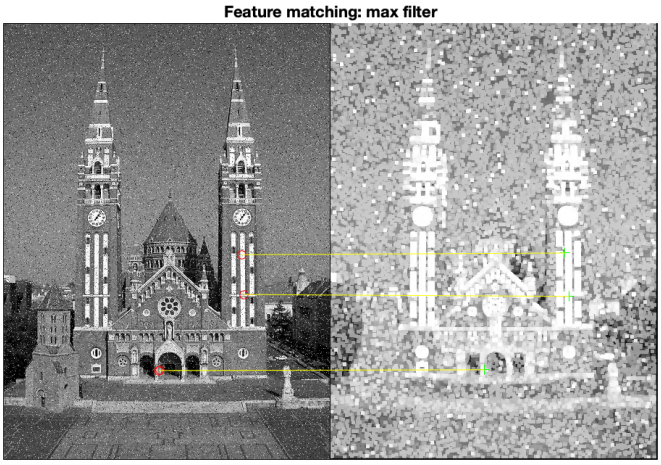



**Question 2.2** *Discuss your observations.*

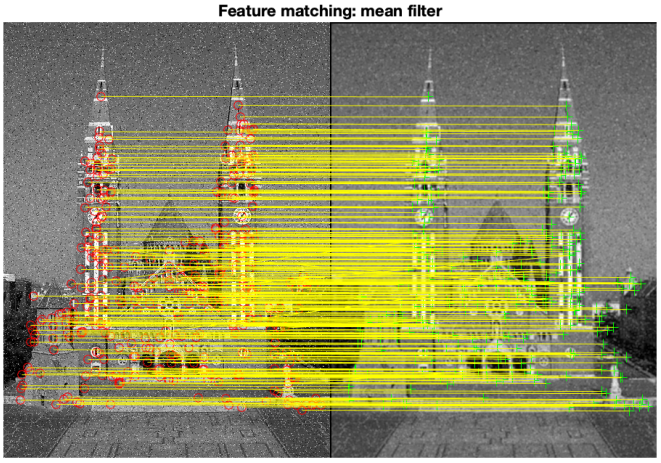
For the Minimum denoising, we can see most features are not matched between the two images. It is slightly different each time the operation is run, as the Salt  Pepper noise is randomised, but there are rarely exact matches. Here there was a decent attempt at matching the left edge of the right tower, but it isn't perfect. This shows the minimum filter erased most of the details.



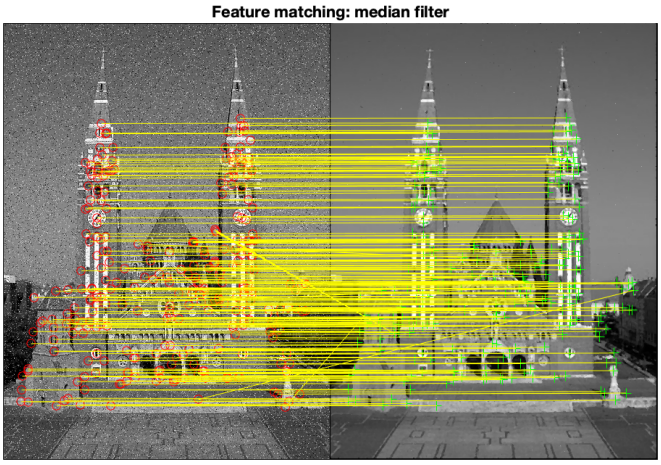

For the Maximum denoising, similarly to minimum denoising, most features are lost. Again, the results are different each time but overall there are very few correct matches. Here the details on the right clock were decently matched, as well as the entrance archway, but that is all. This shows the maximum filter erased most of the details.




For the Mean denoising, most features are correctly matched. We can see some lines are slightly non-parallel to the other ones, and sometimes there is an incorrect match, but overall most features are preserved. This shows the mean filter doesn't erase most of the details.




For the Median denoising, similarly to Mean denoising, most features are correctly matched. Again, some features are incorrectly matched as shown by the lines that aren't parallel to the other ones, but overall most features are preserved. This shows the median filter also successfully preserves most of the details.



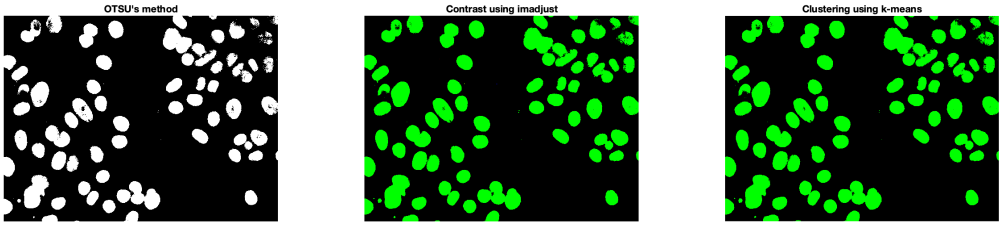

# Part 3

For this task, you will have to submit the following file:

- code **username_assignment1_part3.m**

**Question 3.1** *Compare your results to the results of using the OTSU's method and comment on the reasons for the observed differences in results. If other methods produce worse results than the default OTSU's method, then please explain why you think this is.*

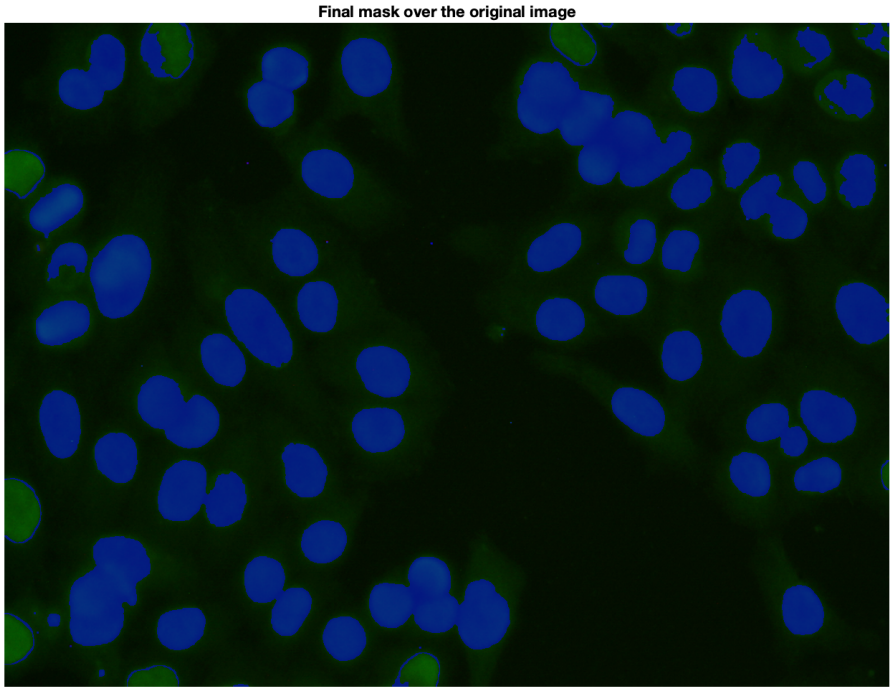
I did not successfully separate each cell, with some clusters remaining, and therefore these results are not fully accurate. Based on estimation, I compared the number of cells (including clusters of unseparated cells as 1 cell) and average area of the cells to choose which method reduce the amount of noise and debris the most. I first used OTSU's method, using the Graythresh function. This method worked well and was easy to implement. Visually, I could see some specks left over, and some of the debris could be confused with separate cells. At the end of my full processing, I was left with 83 'cells' and an average cell area of 2.7853e+03 pixels.
I then tested Multithresh, but the results were extremely similar to Graythresh.
Next, I tried manually modifying the contrast using Imadjust. I specified 'low in' and 'high in' values, and found the most conclusive values to be 0.19 and 0.21. Visually, the results were very similar to Graythresh. However at the end of the full processing, only 75 'cells' were counted, with an average area of 3.1890e+03 pixels. This means this method was finding fewer separate elements and they had a larger value, while no visible cells were being omitted, therefore some of the noise that was being counted as elements with Graythresh was eliminated.
Lastly, I tested k-means clustering, however the results were similar to my previous option, and the format of the result was inconvenient for further processing.
I finally decided to proceed using Imadjust to segment the cells.




**Question 3.2** *Explain the purpose and the outcome(s) of any operation that you have used.*

I first outlined each cell using the Sobel operator. This gives me a mask of the outline of each cell, as well as some noise and debris.
Next, I dilated the image to ensure all the edges detected were connected. This provided full outlines for each cell.
I then filled the interior gaps, so that the whole cell is selected and not just its outline. This caused some cells to be omitted, notably those that weren't fully in the picture, as they had not closed edge to be filled.
Lastly, I eroded the image to smooth out the irregularities. This removed some of the noise around the cells, and made the outline resemble more closely the smooth edges of the cells in the original image.

When overlapping the final mask with the original image, I found that although some cells were omitted, the mask is close enough to the original sizes and placements of the cells.




**Question 3.3.1** *Area (in pixels) of each cell*
Please refer to the attached CSV file.

**Question 3.3.2** *Mean brightness (in green channel) of each cell*
Please refer to the attached CSV file.

**Question 3.3.3** *Mean and standard deviation for the area and brightness for all the cells in the image*

| | |
|---|---|
| Mean area | 1.7669e+03 pixels |
| Standard deviation | 3.1222e+03 pixels |
| Mean brightness | 77.5 |
| Standard deviation | 44.6001 |